In *RoboCup 2003: Robot Soccer World Cup VII, LNCS Vol. 3020*
Daniel Polani, et al., eds, pages 43-53, Springer-Verlag, Berlin Heidelberg, 2004

On role allocation in RoboCup*

Brian P. Gerkey[1] and Maja J Matarić[2]

[1] Robotics Laboratory,
Stanford University, Stanford CA 94305, USA,
gerkey@robotics.stanford.edu,
http://robotics.stanford.edu/~gerkey
[2] Computer Science Department,
University of Southern California, Los Angeles CA 90089, USA,
mataric@cs.usc.edu,
http://robotics.usc.edu/~maja

Abstract. A common problem in RoboCup is *role allocation*: given a team of players and a set of roles, how should be roles be allocated to players? Drawing on our previous work in multi-robot task allocation, we formalize the problem of role allocation as an iterated form of *optimal assignment*, which is a well-studied problem from operations research. From this perspective, we analyze the allocation mechanisms of a number of RoboCup teams, showing that most of them are greedy, and that many are in fact equivalent, as instances of the canonical Greedy algorithm. We explain how *optimal*, yet *tractable*, assignment algorithms could be used instead, but leave as an open question the actual benefit in terms of team performance of using such algorithms.

1 Introduction

Over the past decade, a significant shift of focus has occurred in the field of mobile robotics as researchers have begun to investigate problems involving multiple, rather than single, robots. From early work on loosely-coupled tasks such as homogeneous foraging (e.g., [1]) to more recent work on team coordination for robot soccer (e.g., [2]), the complexity of the multi-robot systems being studied has increased. This complexity has two primary sources: larger team sizes and greater heterogeneity of robots and tasks. As significant achievements have been made along these axes, it is no longer sufficient to show, for example, a pair of robots observing targets or a large group of robots flocking as examples of coordinated robot behavior. Today we reasonably expect to see increasingly larger robot teams engaged in concurrent and diverse tasks over extended periods of time.

As a result of the growing focus on multi-robot systems, multi-robot coordination has received significant attention. In particular, *multi-robot task allocation*

* This paper is based in part on "A Framework for Studying Multi-Robot Task Allocation" by B.P. Gerkey and M.J Matarić, appearing in Schultz, A., et al., eds.: *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, Kluwer Academic Publishers, the Netherlands (2003) 15–26.

(MRTA) has recently risen to prominence and become a key research topic in its own right. As researchers design, build, and use cooperative multi-robot systems, they invariably encounter the fundamental question: "which robot should execute which task?" in order to cooperatively achieve the global goal.

In the RoboCup domain, this problem is usually referred to as *role allocation*, with the concept of a time-extended "role" replacing that of a transient "task". Regardless, the underlying problem remains the same, so we will use the terms task and role interchangeably throughout the paper. As in most other multi-robot systems, the question of how to assign roles in RoboCup is of significant importance for most teams, as it forms the very core of their teamwork strategy[3]. However, the methods demonstrated to date remain primarily *ad hoc* in nature, and relatively little has been written about the general properties of the role allocation problem. The field still lacks a prescription for how to design a role allocation mechanism and there has been little attempt to analytically evaluate or compare the proposed techniques. They have, of course, been extensively evaluated *empirically*.

In this paper we attempt to fill this gap with a formal discussion of role allocation in RoboCup. Our goal is to provide a framework in which to understand the underlying problem, as well as existing solutions. solutions. We show how role allocation can be cast as an iterated form of the long-studied Optimal Assignment Problem (OAP) [4]. In this light, we find that many allocation mechanisms are in fact algorithmically equivalent, as instances of the canonical Greedy algorithm, and thus have known worst-case bounds on solution quality. We also explain how *provably optimal* allocation techniques could be used instead, with a negligible increase in computational overhead. We leave as an open question the actual performance benefit of choosing an optimal allocation algorithm over a greedy one.

The rest of this paper is organized as follows. In the next section, we formalize the problem of role allocation as it is encountered in RoboCup, and discuss greedy and optimal algorithms for solving this problem. In Section 3, we provide a number of examples from the RoboCup literature of teams performing role allocation, and analyze their algorithms in the context of our formalization. We conclude in Section 4 with a discussion of future directions for this kind of analysis in multi-robot coordination problems.

2 The problem

When studying the problem of multi-robot role allocation we take inspiration from operations research, a field that concerns itself with human organizations. In particular we claim that role allocation can be reduced to an iterated form of the *Optimal Assignment Problem* (OAP) [4], a well-known problem from operations research. A recurring special case of particular interest in several fields of study,

[3] Of course, not all teams use role-based strategies, nor do all role-based teams employ *explicit* role assignment (an example of *intentional* cooperation [3]); we restrict our attention in this paper to those teams that do both.

this problem can be formulated in many ways. Given our application domain, it is fitting to describe the problem in terms of jobs and workers. There are n workers, each looking for one job, and n available jobs, each requiring one worker. The jobs can be of different priorities, meaning that it is more important to fill some jobs than others. Each worker has a nonnegative skill rating estimating his/her performance for each potential job (if a worker is incapable of undertaking a job, then the worker is assigned a rating of zero for that job). The problem is to assign workers to jobs in order to maximize the overall expected performance, taking into account the priorities of the jobs and the skill ratings of the workers. This problem was first formally studied in the context of assigning naval personnel to jobs based on the results of aptitude tests [5].

Our multi-robot role allocation problem can be posed as an assignment problem in the following way: given n robots, n prioritized (i.e., weighted) single-robot role, and estimates of how well each robot can be expected to play each role, assign robots to roles so as maximize overall expected performance. However, because the problem of role allocation is a dynamic decision problem that varies in time with phenomena including environmental changes, we cannot be content with this static assignment problem. Thus we complete our reduction by iteratively re-solving the static assignment problem over time.

Of course, the cost of running the assignment algorithm must be taken into account. At one extreme, a costless algorithm can be executed arbitrarily quickly, ensuring an efficient assignment over time. At the other extreme, an expensive algorithm that can only be executed once will produce a static assignment that is only initially efficient and will degrade over time. Finally there is the question of how many roles are considered for (re)assignment at each iteration. In order to create and maintain an efficient allocation, the assignment algorithm must consider (and potentially reassign) every role in the system. Such an inclusive approach can be computationally expensive and, indeed, some implemented approaches to role allocation use heuristics to determine a subset of roles that will be considered in a particular iteration.

Together, the cost of the static algorithm, the frequency with which it is executed, and the manner in which roles are considered for (re)assignment will determine the overall computational and communication overhead of the system, as well as the solution quality. Before continuing with a formal statement of the role assignment problem as an instance of OAP, we first introduce the vital concept of *utility*.

2.1 Utility

Utility is a unifying, if sometimes implicit concept in economics, game theory, and operations research, as well as multi-robot coordination. The underlying idea in all fields is that each individual can somehow internally estimate the value (or the cost) of executing an action. It is variously called fitness, valuation, and cost. Within multi-robot research, the formulation of utility can vary from sophisticated planner-based methods [6] to simple sensor-based metrics [7]. In the RoboCup domain, it is common to compute utility as the weighted some

of several factors, such as distance to a target position, distance from the ball, whether the team is on offense or defense, etc. We posit that utility estimation of this kind is carried out somewhere in every autonomous task or role allocation system.

Regardless of the method used for calculation, the robots' utility estimates will be inexact for a number of reasons, including sensor noise, general uncertainty, and environmental change. These unavoidable characteristics of the multi-robot domain will necessarily limit the efficiency with which coordination can be achieved. We treat this limit as exogenous, on the assumption that lower-level robot control has already been made as reliable, robust, and precise as possible and thus that we are incapable of improving it. We discuss "optimal" allocation solutions in the sense that, under the assumption that all information available to the system (with the concomitant noise, uncertainty, and inaccuracy) is contained in the utility estimates, it is impossible to construct a solution with higher overall utility; this notion of optimality is analogous to optimal scheduling [8].

It is important to note that utility is an extremely flexible measure of fitness, into which arbitrary computation can be placed. The only constraint on utility estimators is that they must each produce a single scalar value such that they can be compared for the propose of ordering candidates for tasks. For example, if the metric for a particular task is distance to a location and the involved robots employ a probabilistic localization mechanism, then one reasonable utility estimator would be to calculate the center of mass of the current probability distribution. Other mechanisms, such as planning and learning, can likewise be incorporated into utility estimation. No matter the domain, it is vital that *all* relevant aspects of the state of the robots and their environment be included in the utility calculation. Signals that are left out of this calculation but are taken into consideration when evaluating overall system performance are what economists refer to as *externalities* [9]; their effects can be detrimental, if not catastrophic.

2.2 Formalism

We are now ready to state our role allocation problem as an instance of the OAP. Formally, we are given:

- the set of n robots, denoted $I_1, \ldots, I_n$
- the set of n prioritized roles, denoted $J_1, \ldots, J_n$ and their relative weights $w_1, \ldots, w_n$
- U_{ij}, the nonnegative utility of robot I_i for role J_j, $1 \leq i, j \leq n$

We assume:

- Each robot I_i is capable of executing at most one role at any given time.
- Each role J_j requires exactly one robot to execute it.

These assumptions, though somewhat restrictive, are necessary in order to reduce role allocation to the classical OAP, which is given in terms of single-worker jobs

and single-job workers. It is worth noting that in most existing role allocation work, especially in RoboCup, these same assumptions are made.

The problem is to find an optimal allocation of robots to roles. An allocation is a set of robot-role pairs:

$$(i_1, j_1) \dots (i_n, j_n)$$

Given our assumptions, for an allocation to be *feasible* the robots $i_1 \dots i_n$ and the roles $j_1 \dots j_n$ must be unique. The benefit (i.e., expected performance) of an allocation is the weighted utility sum:

$$U = \sum_{m=1}^{n} U_{i_m j_m} w_{j_m}$$

We can now cast our problem as an integral linear program [4]: find n^2 nonnegative integers α_{ij} that maximize

$$\sum_{i,j} \alpha_{ij} U_{ij} w_j \tag{1}$$

subject to

$$\begin{aligned} \sum_{i} \alpha_{ij} &= 1, \forall j \\ \sum_{j} \alpha_{ij} &= 1, \forall i \end{aligned} \tag{2}$$

The sum (1) is just the overall system utility, while (2) enforces the constraint that we are working with single-robot roles and single-role robots (note that since α_{ij} are integers they must all be either 0 or 1). Given an optimal solution to this problem (i.e., a set of integers α_{ij} that maximizes (1) subject to (2)), we construct an optimal role allocation by assigning robot i to role j only when $\alpha_{ij} = 1$.

By creating a *linear* program, we restrict the space of role allocation problems that we can model in one way: the function to be maximized (1) must be linear. Importantly, there is no such restriction on the manner in which the components of that function are derived. That is, individual utilities can be computed in any arbitrary way, but they must be combined linearly.

2.3 Solutions

If the robots' utilities can be collected at one machine (or distributed to all machines), then a centralized allocation mechanism can be employed. This is often the case in RoboCup, where locally-computed utility values are generally either unicast to a single player/coach or broadcast to all players (so that they each have the same data). In the former case, one machine can execute the

allocation algorithm and inform the players of the results; in the latter, all players can execute the allocation algorithm in parallel[4].

Perhaps the most common role allocation technique is the following

1. Assemble the utility values into an $n \times n$ matrix.
2. Find the highest utility u_{ij}, assign robot i to role j, and cross out row i and column j from the utility matrix.
3. Repeat step 2 until all roles have been assigned.

Intuitively, this technique is greedy, in that it always selects the next best choice. In fact, this technique is an instance of the canonical Greedy algorithm, studied in several areas of optimization [10]. Unfortunately, the OAP does not satisfy the *greedy choice property* [11] and so the Greedy algorithm will not necessarily produce an optimal solution [5]. The worst-case performance of the Greedy algorithm on the OAP is well-known: it is 2-competitive [13]. An algorithm is said to be α-competitive if, for any input, it finds a solution that is no worse than $\frac{1}{2}$ of the optimum. Thus for an assignment problem, the Greedy algorithm will in the worst case produce a solution with utility that is $\frac{1}{2}$ of that given by an optimal solution.

In place of this greedy approach, it is possible to employ *optimal* solutions, a great many of which can be found in the literature (for a representative list, see [14]). The best-known approach is the Hungarian method [15], a linear programming algorithm that exploits the special structure of the OAP. This algorithm will find the optimal solution to a role allocation problem in $O(n^3)$ time. We have demonstrated empirically [16] that the constant factor for the Hungarian method is so small that this algorithm could easily be used for real time role allocation in RoboCup teams, where $n \leq 11$.

It is also possible to solve assignment problems in a completely distributed fashion. An optimal distributed algorithm that is relevant to role allocation in the RoboCup domain was derived by viewing the OAP as a "stable marriage" problem [17]: given a group of n boys and a group of n girls, each with preferences (i.e., utilities) over the members of the opposite sex, find a set of pairings such that there exists no boy and girl pair that is not paired together but prefers each other to their current mates. Gale and Shapley developed an intuitive algorithm in which each boy proposes to his favorite available girl and each girl conditionally accepts her favorite proposer. This process is repeated in a series of stages until, when the last girl has received a proposal, the "courtship" is declared to be over and the result is an optimal solution to the problem of assigning boys to girls so as to maximize total utility.

This distributed algorithm obviates the need to communicate the individuals' utilities among the individuals and requires $O(n^2 - 2n + 2)$ stages, which may

[4] Due to communication issues, different players may occasionally have different utility information. Because reallocation is performed so frequently, the effects of such inconsistency are usually transient and negligible.

[5] Equivalently, the OAP is a maximization problem over a subset system, but is not a *matroid* [12].

seem better than the $O(n^3)$ running time for centralized solutions. Unfortunately for high-speed, real-world domains like RoboCup, each stage in the proposal algorithm requires a synchronous exchange of messages between some or all of the participants. Given the latency characteristics of real networks, especially wireless networks in congested environments, the time required at each stage to guarantee the receipt of all messages and maintain synchronization makes this algorithm far too slow.

In practice, a team seeking optimal role allocation will be better off periodically broadcasting utility values to all players and having the players run the same centralized assignment algorithm in parallel. Assuming that the robots are linked with a wireless network and that a robot's utility values for all roles can be included in a single packet (this is almost certainly the case, as utilities are scalars), the communication cost of informing n robots of each others' utilities is just n messages. This overhead is negligible, as is the ensuing computational cost of executing even an optimal assignment algorithm.

3 Some examples

Inspired by the pioneering work of Stone and Veloso [2], many RoboCup teams employ role-based coordination, in which the players can take on different roles within the team. Although it is possible to statically assign roles once at the beginning of the game, this strategy is brittle and unable to exploit unexpected opportunities [18]. Thus most role-based teams employ some kind of dynamic role allocation, in which the current allocation is reevaluated periodically, usually on the order of 10Hz. Thus they are solving the iterated assignment problem, though often without recognizing it as such (a notable exception is RoboLog Koblenz [19], who identify the problem of assigning players to block opponents as one of stable marriage, though it is not clear whether they actually use an optimal algorithm to solve it). Furthermore, most teams use greedy algorithms, despite the fact that tractable optimal algorithms exist.

For example, the assignment algorithm of ART99 consists of ordering the roles in descending priority and then assigning each to the available robot with the highest utility [20]. This algorithm is clearly an instance of the Greedy algorithm, and thus is 2-competitive. As is the case with most teams, utilities in ART99 are computed in a role-specific manner. The same approach, with utility referred to as *function Q*, is used in [21]. Vail and Veloso also employ the Greedy algorithm, with fixed priority roles [22]. Another team [23] describes a more complex approach in which role allocation is carried out differently depending on the game situation, such as who has the ball. We conjecture that their technique can be reduced to the Greedy algorithm by combining the different cases and appropriately modifying the utility functions.

A common problem with dynamic role assignment is that small changes in utility estimates can cause roles to be reassigned very frequently, often in an oscillating fashion. Changing from one role to another is not a costless operation; for example, a player may have to move from its current position to another

position on the field. If the team is constantly reassigning roles it will spend less time actually playing and will tend not to perform well. The underlying problem is that the cost of transitioning from one role to another is an externality: it affects system performance but is not included in the utility computation. Thus the solution is to explicitly consider the cost of role transition in the utility computation, which will tend to induce a degree of hysteresis. Some teams, such as RMIT United [24], do this by adding a fixed amount to the utility of a robot retaining its current role

While the teams discussed so far employ centralized assignment algorithms, distributed role-swapping approaches are also used. The winners of the F2000 league at RoboCup 2000, CS Freiburg [25], use a distributed role allocation mechanism in which two players may exchange roles only if both "want" to, in that they will both be moving to higher-utility roles for themselves. This algorithm is similar to the marriage proposal algorithm discussed previously, but is not optimal, a fact that can easily be shown by counterexample. This result is not surprising because an optimal role-swapping algorithm, which considers all feasible assignments, would be too slow in practice. A similar pairwise exchange mechanism is used by FC Portugal [26], who won the simulation league at RoboCup 2000.

Unfortunately, without more detailed information about these role-exchange algorithms (e.g., how often are exchanges attempted? how many are attempted?), it is impossible to derive any performance bounds, other than to say that there exist situations in which they will produce sub-optimal solutions. Furthermore, we conjecture that little is saved in terms of communication or computation by executing such a distributed allocation algorithm. Instead of pursuing pairwise exchanges, the robots could periodically broadcast their utility values for all roles, then each execute an optimal centralized assignment algorithm.

4 Discussion

In this paper we have presented a formal framework for studying the RoboCup role allocation problem. We have shown how this problem can be understood as an iterated Optimal Assignment Problem (OAP), and how many existing team coordination strategies can be seen as assignment algorithms that are used to solve this problem. We have also explained how tractable optimal assignment algorithms developed in operations research can be immediately applied to RoboCup role allocation problems, in place of the existing greedy and/or *ad hoc* solutions.

A natural question arises: what will actually be gained by employing an optimal solution? Consider the analysis showing that the allocation mechanism used by ART99, as an implementation of the canonical Greedy algorithm, is 2-competitive. This kind of competitive factor gives an algorithm's *worst-case* behavior, which may be quite different from its *average-case* behavior. In fact, for many small assignment problems, the Greedy algorithm will find an optimal or near-optimal solution. In this respect, the solution quality bounds established

for existing allocation architectures are rather loose. One way to tighten these bounds is to add domain-specific information to the formalism. By capturing and embedding models of how real RoboCup teams behave and evolve over time, it should be possible to make more accurate predictions about algorithmic performance. For example, while the classical theory of the OAP makes no assumptions about the nature of the utility matrices that form the input, role allocation problems are likely to exhibit significant structure in their utility values. Far from randomly generated, utility values generally follow one of a few common models, determined primarily by the kind of sensor data that are used in estimating utility. If only "local" sensor information is used (e.g., can the robot currently see the ball, and if so, how close is it?), then utility estimates tend to be strongly bifurcated (e.g., a robot will have very high utility if it can see the ball, and zero utility otherwise). On the other hand, if "global" sensor information is available (e.g., how close is the robot to a goal position?), then utility estimates tend to be smoother (e.g., utility will fall off smoothly in space away from the goal). A promising avenue for future research would be to characterize this "utility landscape" as it is encountered in RoboCup, and make predictions about, for example, how well a greedy assignment algorithm should be expected to work, as opposed to a more costly optimal assignment algorithm.

Another important issue concerns the way in which the role allocation problem is traditionally framed. Throughout this paper, we have considered the problem of assigning a single robot to a single role. In the context of our previously developed taxonomy of such problems [16], we are studying an iterated instance of ST-SR-IA (single-task robots, single-robot tasks, instantaneous assignment). However, in some situations, such as when two or more defenders are needed to block an attacker, the problem actually involves assigning *multiple* robots to a single role or task. It is generally not the case that the utility of a collection of robots for a given task will be equal to the sum of their individual utilities. Thus the classical assignment formulation is only an approximation of the real problem, which is an instance of ST-MR-IA (same as ST-ST-IA, but with multi-robot tasks). Unfortunately, problems of this kind, which involve *coalition formation*, are rather difficult to solve. In the most general case, ST-MR-IA problems are a form of set partitioning, which is strongly NP-hard [27]. As such, using the single-robot task approximation is a parsimonious choice.

We have also assumed thus far that robots' utilities are *independent*. That is, a robot's utility for a particular role does not depend on the other robots' allocated roles. This assumption is rarely, if ever, true in multi-robot systems. In general, a robot's utility for a role can in fact be affected by the overall allocation of roles to robots. This situation can arise any time that physical interference contributes significantly to task performance [28]. Unfortunately, the state of the art for capturing interrelated utilities of this kind is the Markov Decision Process [29], which remain too difficult to solve quickly enough for domains like RoboCup (especially if partial observability is considered).

Because it is well-defined and requires researchers to empirically compare their proposed techniques, RoboCup is an excellent domain in which to study

role allocation. It is our hope that the formal perspective on the problem that we have given in this paper will aid in the understanding of how and why existing techniques work, as well as guide the design of new ones.

References

1. Matarić, M.J.: Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In Meyer, J.A., Roitblat, H., Wilson, S., eds.: From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92), MIT Press (1992) 432–441
2. Stone, P., Veloso, M.: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. Artificial Intelligence **110** (1999) 241–273
3. Parker, L.E.: ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. IEEE Transactions on Robotics and Automation **14** (1998) 220–240
4. Gale, D.: The Theory of Linear Economic Models. McGraw-Hill Book Company, Inc., New York (1960)
5. Thorndike, R.L.: The Problem of Classification of Personnel. Psychometrika **15** (1950) 215–235
6. Botelho, S.C., Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), Detroit, Michigan (1999) 1234–1239
7. Gerkey, B.P., Matarić, M.J.: Sold!: Auction methods for multi-robot coordination. IEEE Transactions on Robotics and Automation **18** (2002) 758–768
8. Dertouzos, M.L., Mok, A.K.: Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks. IEEE Transactions on Software Engineering **15** (1983) 1497–1506
9. Simon, H.A.: The Sciences of the Artificial. 3rd edn. MIT Press, Cambridge, Massachusetts (2001)
10. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River, New Jersey (1993)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge, Massachusetts (1997)
12. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer-Verlag, Berlin (2000)
13. Avis, D.: A Survey of Heuristics for the Weighted Matching Problem. Networks **13** (1983) 475–493
14. Bertsekas, D.P.: The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial. Interfaces **20** (1990) 133–149
15. Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly **2** (1955) 83–97
16. Gerkey, B.P., Matarić, M.J.: A formal framework for the study of task allocation in multi-robot systems. Technical Report CRES-03-013, Center for Robotics and Embedded Systems, School of Engineering, University of Southern California (2003)
17. Gale, D., Shapley, L.S.: College Admissions and the Stability of Marriage. American Mathematical Monthly **69** (1962) 9–15
18. Howard, A.: MuCows. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 535–538

19. Murray, J., Obst, O., Stolzenburg, F.: RoboLog Koblenz 2001. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup 2001, LNAI 2377. Springer-Verlag, Berlin (2002) 526–530
20. Castelpietra, C., Iocchi, L., Nardi, D., Piaggio, M., Scalzo, A., Sgorbissa, A.: Communication and Coordination among heterogeneous Mid-size players: ART99. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 86–95
21. Ferraresso, M., Ferrari, C., Pagello, E., Polesel, R., Rosati, R., Speranzon, A., Zanette, W.: Collaborative Emergent Actions between Real Soccer Robots. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 297–302
22. Vail, D., Veloso, M.: Dynamic Multi-Robot Coordination. In Schultz, A., et al., eds.: Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II. Kluwer Academic Publishers, the Netherlands (2003) 87–98
23. Jamzad, M., Chitsaz, H., Foroughnassirai, A., Ghorbani, R., Kazemi, M., Mirrokni, V., Sadjad, B.: Basic Requirements for a Teamwork in Middle Size RoboCup. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup 2001, LNAI 2377. Springer-Verlag, Berlin (2002) 621–626
24. Brusey, J., Makies, M., Padgham, L., Woodvine, B., Fantone, K.: RMIT United. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 563–566
25. Weigel, T., Auerback, W., Dietl, M., Dümler, B., Gutmann, J.S., Marko, K., Müller, K., Nebel, B., Szerbakowski, B., Thiel, M.: CS Freiburg: Doing the Right Thing in a Group. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 52–63
26. Reis, L.P., Lau, N.: FC Portugal Team Description: RoboCup 2000 Simulation League Competition. In Stone, P., Balch, T., Kraetzschmar, G., eds.: RoboCup 2000, LNAI 2019. Springer-Verlag, Berlin (2001) 29–40
27. Garey, M.R., Johnson, D.S.: "Strong" NP-Completeness Results: Motivation, Examples, and Implications. J. of the ACM **25** (1978) 499–508
28. Goldberg, D., Matarić, M.J.: Interference as a tool for designing and evaluating multi-robot controllers. In: Proc. of the Natl. Conf. on Artificial Intelligence (AAAI), Providence, Rhode Island (1997) 637–642
29. White, D.J.: Markov decision processes. John Wiley & Sons, Chichester, England (1993)